SpinalHDL 编写的打地鼠游戏

张政镒　20300750021

2023 年 6 月 17 日

目录

1 设计规划

1.1 设计要求

1.1.1 设计内容

设计一个打地鼠游戏机. 本设计在原题要求上进行了一定的扩展, 使得游戏机具有更好的可玩性. 下面是本设计的主要功能:

- 用 HDMI 显示地鼠, 当前的难度以及得分情况.
- 有一个 Start 按钮用于控制游戏开始, 一个 Reset 按钮用于重置游戏.
- 游戏开始后, 会在屏幕上显示地鼠, 地鼠间隔一定的随机时间 T_1 刷新位置并显示, 并持续 T_2 (随难度变化) 时间.
- 游戏分为 3 轮, 每轮游戏结束后若超过规定分数, 则进入下一轮. 在每轮游戏中, 打到地鼠得到的分数随难度增加而增加.
- 游戏结束后, 显示总分.

1.2 设计思路

1.2.1 硬件设计

我采用 SpinalHDL 完成硬件设计部分. SpinalHDL 是一种基于 Scala 的硬件描述语言, 可以生成 VHDL 或 Verilog 文件, 以此与现有的 EDA 工具兼容. 借由 Scala 提供的现代高级编程语言的特性, SpinalHDL 可以大大提高硬件设计的效率. 它**不是**一种高层次综合语言, 也非基于事件驱动的范式. 它依然是一种基于寄存器传输级 (RTL) 的描述范式.

我采用 SpianlHDL 的原因在于其适用于快速开发迭代流程, 并提供良好的封装和代码可读性. 可以将各个组件低耦合地设计, 并在最后组合起来. 它还提供了强大的参数化设计能力, 而这一点是在 Verilog 中很难做到的.

```
1  import spinal.core._
2  import spinal.lib._
3
4  case class Adder(width: Int) extends Component {
5    val io = new Bundle {
6      val a = in UInt (width bits)
7      val b = in UInt (width bits)
8      val c = out Bool (width bits)
9    }
10
11   io.c := io.a + io.b
12 }
```

Listing 1: SpinalHDL 实现的加法器

如代码?? 所示是一个简单的 SpinalHDL 示例, 其实现了一个位宽参数化的加法器.

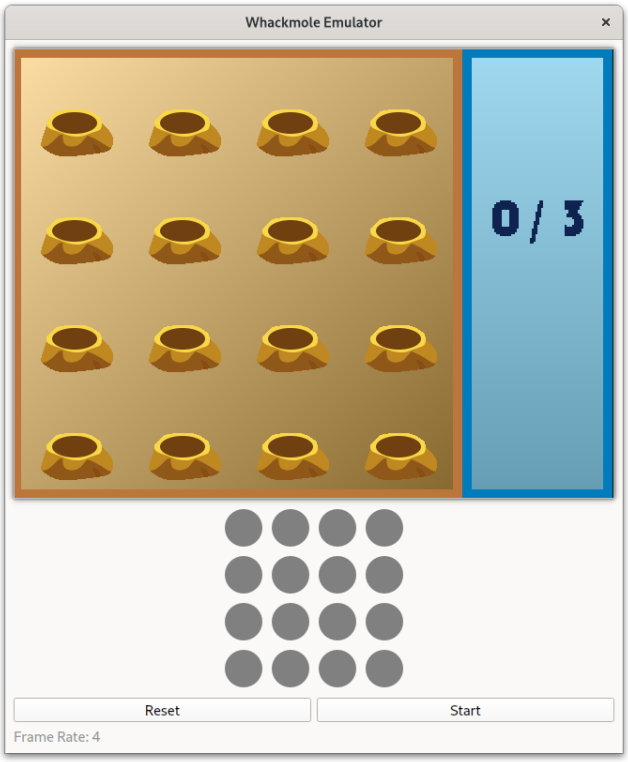

图 1: 以 Verilator 为基础的软件仿真器

1.2.2 软件仿真

我使用了 Verilator 作为基本仿真工具并结合 Qt 作为图形化界面. Verilator 是一个开源的 Verilog 仿真器, 它可以将 Verilog 代码编译成 C++ 代码, 并提供了一套 C++ API 用于仿真. 其优点在于仿真速度快, 且可以方便地与 C++ 代码进行交互. Qt 是一个跨平台的 GUI 库, 可以方便地实现图形化界面. 通过读取 Verilator 生成的仿真结果, 我可以在 C++ 中读取每一个周期的 RGB, VSYNC, HSYNC 信号, 并将其进行拼接, 最终在软件端模拟一个显示器进行显示. 仿真器如图 **??** 所示.

显然, 软件端的模拟不可能达到硬件端的速度. 我采用的是 640x480 的分辨率, 以 60Hz 的刷新率进行显示的时序. 然而, 软件端只能做到以 10Hz 的刷新率进行显示. 为了保证软件运行的稳定性 (过高的刷新率容易导致处理跟不上, 从而使得跨线程信号堆积), 我把软件的刷新率固定在了 5Hz. 那么, 我们怎么保证在时钟频率不同的硬件端和软件端游戏体验一致呢?

好在, SpinalHDL 为我们提供了一套封装, 让我可以使用" 秒" 和" 频率" 作为参数进行设计. 如代码 **??** 所示. 这段代码是我实际使用的代码.

```
1 val frequency: HertzNumber = 2 MHz,
2 val roundGapTime: TimeNumber = 3 sec,
```

Listing 2: SpinalHDL 中的时间参数

我向软件端传输的是 VGA 信号, 则刷新一帧需要 $525 \times 800 = 420000$ 个周期. 在 2 MHz 的像素时钟下, 一帧需要 $420000/2000000 = 0.21$ 秒. 这样就能适应软件端 5 Hz 的刷新率.

如果要在硬件端实现, 我只需要将 2 MHz 改变为标准时钟频率 25.175 MHz 即可. 所有的时间参数, 例如各轮之间的时长都不需要做任何额外的代码修改就能保持一致.

1.3 设计流程图及 EDA 使用说明

2 设计实现

2.1 框图介绍

2.2 各模块设计与验证

2.3 综合与实现

2.4 综合结果分析

2.5 编程工具与下载测试

3 设计总结

3.1 注意事项与编程技巧

3.2 心得体会

3.3 总结

表 1: 主要文件列表

文件名	路径	说明
main.scala	src/main/scala	主文件

A 主要文件列表